In [16]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
from torch.utils.data import DataLoader, random_split
import torchvision.models as models
import torchvision.transforms as transforms
from custom_dataset import RetinaDataset
from sklearn.metrics import accuracy_score
import copy
from copy import deepcopy

# Data augmentation and normalization for training
# Just normalization for validation
train_transforms = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(20),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

val_transforms = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

# Load datasets
dataset = RetinaDataset(csv_file='resized_train/train.csv', root_dir='resized_t
train_size = int(0.7 * len(dataset))
val_size = int(0.15 * len(dataset))
test_size = len(dataset) - (train_size + val_size)
train_dataset, val_dataset, test_dataset = random_split(dataset, [train_size, v
train_dataset.dataset.transform = train_transforms

# Data loaders
batch_size = 32
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, n
val_loader = DataLoader(val_dataset, batch_size=batch_size, num_workers=4)
test_loader = DataLoader(test_dataset, batch_size=batch_size, num_workers=4)

# Load a pretrained model and reset final fully connected layer
model = models.vgg16(pretrained=True)
for param in model.features.parameters():
    param.requires_grad = False

# Parameters of newly constructed modules have requires_grad=True by default
num_ftrs = model.classifier[6].in_features
model.classifier[6] = nn.Sequential(
    nn.Linear(num_ftrs, 4096),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(4096, 4096),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(4096, 5)
)

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)

# Training Loop
best_val_loss = float('inf')
epochs = 25
for epoch in range(epochs):
    model.train()
    running_loss = 0.0
    running_corrects = 0

    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        _, preds = torch.max(outputs, 1)
        running_corrects += torch.sum(preds == labels.data)

    epoch_loss = running_loss / len(train_loader)
    epoch_acc = running_corrects.double() / len(train_loader.dataset)

    # Validation
    model.eval()
    val_loss = 0.0
    val_corrects = 0
    with torch.no_grad():
        for images, labels in val_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            loss = criterion(outputs, labels)
            val_loss += loss.item()
            _, preds = torch.max(outputs, 1)
            val_corrects += torch.sum(preds == labels.data)

    val_loss = val_loss / len(val_loader)
    val_acc = val_corrects.double() / len(val_loader.dataset)

    print(f'Epoch {epoch+1}/{epochs}, '
          f'Training Loss: {epoch_loss:.4f}, Acc: {epoch_acc:.4f}, '
          f'Validation Loss: {val_loss:.4f}, Acc: {val_acc:.4f}')

    # Deep copy the model if it's the best so far
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        best_model_wts = copy.deepcopy(model.state_dict())
    # Step the scheduler
    scheduler.step()

# Load best model weights
model.load_state_dict(best_model_wts)
```

```
Initializing RetinaDataset with extension support

C:\Python311\Lib\site-packages\torchvision\models\_utils.py:208: UserWarning:
The parameter 'pretrained' is deprecated since 0.13 and may be removed in the
future, please use 'weights' instead.
  warnings.warn(
C:\Python311\Lib\site-packages\torchvision\models\_utils.py:223: UserWarning:
Arguments other than a weight enum or `None` for 'weights' are deprecated sin
ce 0.13 and may be removed in the future. The current behavior is equivalent
to passing `weights=VGG16_Weights.IMAGENET1K_V1`. You can also use `weights=V
GG16_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
```

```
Epoch 1/25, Training Loss: 1.0410, Acc: 0.7169, Validation Loss: 1.0544, Acc:
0.7402
Epoch 2/25, Training Loss: 0.9493, Acc: 0.7279, Validation Loss: 0.9249, Acc:
0.7402
Epoch 3/25, Training Loss: 0.9064, Acc: 0.7304, Validation Loss: 0.8227, Acc:
0.7402
Epoch 4/25, Training Loss: 0.8891, Acc: 0.7304, Validation Loss: 0.8132, Acc:
0.7402
Epoch 5/25, Training Loss: 0.8797, Acc: 0.7304, Validation Loss: 0.8491, Acc:
0.7402
Epoch 6/25, Training Loss: 0.8748, Acc: 0.7312, Validation Loss: 0.8267, Acc:
0.7402
Epoch 7/25, Training Loss: 0.8706, Acc: 0.7312, Validation Loss: 0.8185, Acc:
0.7402
Epoch 8/25, Training Loss: 0.8519, Acc: 0.7316, Validation Loss: 0.8000, Acc:
0.7402
Epoch 9/25, Training Loss: 0.8482, Acc: 0.7316, Validation Loss: 0.8043, Acc:
0.7402
Epoch 10/25, Training Loss: 0.8419, Acc: 0.7315, Validation Loss: 0.7952, Ac
c: 0.7402
Epoch 11/25, Training Loss: 0.8337, Acc: 0.7314, Validation Loss: 0.7908, Ac
c: 0.7402
Epoch 12/25, Training Loss: 0.8191, Acc: 0.7315, Validation Loss: 0.7769, Ac
c: 0.7402
Epoch 13/25, Training Loss: 0.8101, Acc: 0.7336, Validation Loss: 0.7703, Ac
c: 0.7416
Epoch 14/25, Training Loss: 0.8031, Acc: 0.7323, Validation Loss: 0.7746, Ac
c: 0.7413
Epoch 15/25, Training Loss: 0.7967, Acc: 0.7352, Validation Loss: 0.7721, Ac
c: 0.7416
Epoch 16/25, Training Loss: 0.7940, Acc: 0.7354, Validation Loss: 0.7683, Ac
c: 0.7413
Epoch 17/25, Training Loss: 0.7970, Acc: 0.7359, Validation Loss: 0.7666, Ac
c: 0.7410
Epoch 18/25, Training Loss: 0.7947, Acc: 0.7369, Validation Loss: 0.7671, Ac
c: 0.7413
Epoch 19/25, Training Loss: 0.7917, Acc: 0.7358, Validation Loss: 0.7691, Ac
c: 0.7418
Epoch 20/25, Training Loss: 0.7936, Acc: 0.7348, Validation Loss: 0.7696, Ac
c: 0.7416
Epoch 21/25, Training Loss: 0.7881, Acc: 0.7371, Validation Loss: 0.7697, Ac
c: 0.7418
Epoch 22/25, Training Loss: 0.7852, Acc: 0.7373, Validation Loss: 0.7671, Ac
c: 0.7418
Epoch 23/25, Training Loss: 0.7883, Acc: 0.7370, Validation Loss: 0.7672, Ac
c: 0.7416
Epoch 24/25, Training Loss: 0.7897, Acc: 0.7364, Validation Loss: 0.7703, Ac
c: 0.7418
Epoch 25/25, Training Loss: 0.7887, Acc: 0.7361, Validation Loss: 0.7706, Ac
c: 0.7416
```

```
Out[16]: <All keys matched successfully>
```

```
In [ ]:
```